GraphWalker: Walk-and-Rephrase Agentic LLM Interplay for Robust and Grounded Knowledge Graph Question Answering

Journal Title
XX(X):1–13
©The Author(s) 2026
Reprints and permission:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

Michael Weyns[1], Bram Steenwinckel[2], Bart Matthys[3], Tanguy Coenen[3], and Femke Ongenae[1]

Abstract

Large Language Models (LLMs) have increased the performance of knowledge graph (KG) question answering (QA) significantly, but struggle to jointly guarantee factual grounding, traceable reasoning and robust multi-hop inference. Hallucinated answers and vague reasoning chains are especially problematic when trustworthiness is required. We propose GraphWalker, an agentic LLM paradigm relying on the interplay between two agents leveraging KG interconnections to overcome these issues. The first agent takes a particular node and question as input and operates in the small immediate neighbourhood around the node to make an informed decision about which direction should be explored to gather more relevant context information before formulating an answer. The second agent generates targeted subquestions, based on the main question and the path that has been walked so far, to steer the first agent. This second agent is also responsible for assessing whether enough context has been gathered so far to formulate the answer. Our design, which aims to reduce hallucinations by dynamically exploring only those KG regions relevant to the original question, enforces step-level provenance of retrieved domain knowledge, as the path that was followed to find the answer can be traced explicitly. We evaluate GraphWalker on the text-based MetaQA splits and compare it to other zero-shot LLM-based question answering approaches, including parametric LLMs, text-based RAG, and KG-based RAG. Across all hop distances, GraphWalker achieves higher answer accuracy (+20.87 percentage points on average over best baseline), produces fewer hallucinated responses (-16.47 percentage points on average over best baseline), and better satisfies the requirements of factual grounding and traceability.

Keywords

Agentic LLM, Question Answering, Multi-hop Reasoning, Traceability, Factual Grounding

1 Introduction

Modern question answering (QA) systems are increasingly built around pre-trained Large Language Models (LLMs), which can understand natural language queries and generate fluent, context-aware answers across domains with little or no task-specific supervision Brown et al. (2020); Ouyang et al. (2022). While showcasing remarkable results in terms of answer accuracy on benchmarks, they have also exhibited significant limitations, notably producing hallucinations and unpredictable responses when handling queries beyond their training data Pan et al. (2024). Therefore, there has been increasing emphasis on evaluating other requirements that assess the functioning of QA systems for real-world applications, namely factual grounding, traceability and handling multi-hop questions Tang and Yang (2024); Kim and Yang (2025).

Factual grounding corresponds to the requirement that answers should be based in reliable factual knowledge. LLMs are known to produce hallucinations, which are coherent but unsupported or incorrect statements, even when they appear confident Kazlaris et al. (2025); Varshney et al. (2023). Recent surveys identify hallucination as one of the main obstacles to deploying LLM-based systems in knowledge-intensive scenarios, emphasizing both its prevalence and its impact on user trust Huang et al. (2025b); Sahoo et al. (2024); Ye et al. (2023). Empirical work shows that hallucinations persist even under instruction tuning and explicit truthfulness objectives, motivating methods that check or constrain model outputs against external evidence Vu et al. (2024). Therefore, QA systems rely on Retrieval Augmented Generation (RAG) to reduce hallucinations Arslan et al. (2024). RAG selects sources from a curated Knowledge Base based on their similarity to the question that is posed and feeds these sources to the LLM to construct its answer. KGs have become prevalent to represent this Knowledge Base due to their inherent ability to make the context of the data clear, improving selection Han et al. (2024). However, even when using RAG, LLMs consistently showcase hallucinations Kümmel (2025). Recent detection and mitigation techniques explicitly rely on

[1]IDLab, Ghent University – imec, 9000 Gent, Belgium
[2]Deloitte, 9051 Gent, Belgium
[3]imec – Application level research and algorithms department, 3000 Leuven, Belgium

Corresponding author:
Michael Weyns
Email: Michael.Weyns@UGent.be

knowledge-based answer fact-checking against structured or unstructured sources or by using uncertainty estimates to flag these hallucinations Farquhar et al. (2024); Ceccarelli et al. (2025). All this research underscores that QA systems must be designed so that answers are not only plausible, but also anchored in accessible factual resources.

A second requirement is *traceability*. One should be able to inspect where an answer came from and how it was produced. Traceability has emerged as a key ingredient of trustworthy AI, as it supports debugging, accountability, and human oversight Afroogh et al. (2024). In the context of LLMs, traceability requires that explanations faithfully reflect the underlying reasoning and that users can reconstruct why a particular answer was chosen Herrera (2025); Mathew et al. (2025). More technical work on provenance-enabled explainable AI shows how maintaining explicit provenance traces allows one to backtrack from outputs to the intermediate context and data that supported them Zhang et al. (2024a). In QA settings, this translates into a requirement that systems expose not just final answers, but also the supporting evidence and intermediate steps used to derive them. Without such traces, users cannot easily distinguish grounded answers from hallucinations nor contest or audit system behavior Lecue (2020). While there has been significant work on improving the traceability of LLM-based QA systems Vladika et al. (2024); Matton et al. (2025), open problems still remain around, e.g. the faithfulness of the provided explanations and tracing the individual answers back to specific, trustworthy sources.

A third requirement is robust *multi-hop reasoning*. Many realistic questions are inherently compositional. They require combining multiple facts, constraints, or modalities rather than retrieving a single snippet Mavi et al. (2024). Multi-hop QA benchmarks consistently show that model performance drops as the number of reasoning hops increases Min et al. (2019); Huang et al. (2025a). This degradation is particularly noticeable for LLM-based systems that rely solely on dense retrieval or parametric knowledge, which tend to miss essential intermediate facts or fail to maintain logically consistent reasoning chains Huang et al. (2025a).

All these requirements point to the importance of *structured grounding*. KGs encode entities and relations as explicit edges and paths, offering a natural format for multi-hop reasoning and generating explanations in the form of traversed paths Zhu et al. (2024). KG-based QA and reasoning frameworks have shown that explicit graph paths can increase answer faithfulness, make reasoning steps more transparent, and support richer forms of explanation Lu et al. (2025); Zhao et al. (2025). However, analyses of LLM+KG systems highlight open challenges. Many current methods use KGs only indirectly, e.g. via static subgraph retrieval, precomputed path embeddings, or textual verbalizations, or provide only partial guarantees that reasoning respects the KG structure Ma et al. (2025); Lavrinovics et al. (2025); Amayuelas et al. (2025). In practice, this means that the KG shapes what is retrieved, but not how the model is allowed to reason. The LLM can still assert or assume relations that are not in the graph or even contradict it, hindering factual grounding. As intermediate hops are often compressed into embeddings or to free-text summaries, it remains unclear which triples support which parts of the answer, and whether all required intermediate nodes and relationships for a multi-hop question were used, undermining traceability. As such, these systems fall short of jointly satisfying the requirements of factual grounding, traceability, and reliable multi-hop reasoning.

Simultaneously, a trend towards agentic AI solutions can be noticed that treats LLMs as planners capable of deciding itself which steps are required to reach a certain goal, e.g. answering a question Acharya et al. (2025). The first steps in using agentic AI for QA over KGs have already been taken Jiang et al. (2024); Liu (2025); Panda et al. (2024); Chen et al. (2025a). However, they fall short of constraining the exploration to the KG structure, inefficiently explore the graph or do not perform step-wise reasoning, leaving them unable to realize the requirements in terms of traceability and factual grounding.

In this work, we take the aforementioned requirements as design principles for an agentic AI solution and propose GraphWalker, a constrained agentic framework for multi-hop QA over KGs. Instead of treating KG access as loosely constrained, GraphWalker elevates graph traversal to a first-class operation by relying on the interplay between two LLM agents:

- **LLM-directed walking**: at each iteration, the first LLM can retrieve new information by executing deterministic KG expansions of depth 1 (i.e., one hop) that prioritize graph regions relevant to the question it tries to solve. As such, this LLM operates over a restricted action space of 1-hop deterministic walk steps available around a node in the KG that is given as input to the LLM together with the question.
- **LLM-guided query refinement**: the original question is decomposed into explicit subquestions by the second LLM by taking into account the path that has been walked in the KG so far. These questions are distributed to the first LLM to resolve directly against the KG and the answers are appended to a structured evidence buffer, which is used by the second LLM to assess whether enough context has been gathered to answer the overall question.

By combining directed KG walking with LLM-guided refinement, GraphWalker enables dynamic, query-driven exploration grounded in the KG while preserving explicit, step-level provenance over nodes and edges, satisfying the discussed requirements.

2 Related work

Three major research lines can be discerned in realizing QA over KGs using LLMs.

LLMs with purely parametric memory. Early LLMs, such as GPT-4 and Llama-2, demonstrated strong zero-shot QA performance relying solely on parametric memory. However, systematic evaluation on multi-hop QA revealed that they perform competitively only when the relevant information is small enough to be provided in-context Chakraborty (2024). As the knowledge complexity increases, such models struggle to capture accurate relational aspects and often hallucinate or compose incorrect relations due to missing explicit step-wise evidence.

Classical RAG. To address this limitation, RAG was introduced to ground responses in retrieved documents rather than internal memory Arslan et al. (2024). While highly effective for single-hop lookups in more dynamic contexts, RAG fails to reliably support complex multi-hop reasoning because it requires chaining multiple evidence fragments. Even more advanced LLMs, such as GPT-4, PaLM 2, and Llama-2-70B, often retrieve incorrect or incomplete context and therefore fail on multi-step queries Tang and Yang (2024). More structured retrieval systems such as HiRAG Zhang et al. (2024b) improve granularity via hierarchical decomposition and filtering, yet still operate over unstructured text lacking the explicit relational structure. As emphasized in recent surveys, simply increasing the retrieval quantity or granularity does not eliminate hallucination nor guarantee faithful reasoning paths Huang et al. (2025b). This limitation motivated the incorporation of more structured knowledge.

KG augmented retrieval and structured multi-Hop reasoning. Work on KG-augmented retrieval integrates the explicit graph structure into the reasoning process to improve the factual precision and compositional reasoning. Prompting strategies, such as Connect-the-Entities (CTE) Sağlam (2024), and hybrid architectures combining KG and LLM knowledge Pan et al. (2023) enhance accuracy by grounding reasoning in triple extraction. More advanced systems, like GraphLLM Qiao et al. (2024), use GNN-based decomposition of multi-hop queries, and reasoning-plan approaches Chen et al. (2025b) to provide greater transparency. Yet most of these pipelines retrieve the subgraphs upfront and apply reasoning in a post hoc manner rather than performing explicit traversal to ensure path fidelity. This motivates tighter coupling between KGs and reasoning.

Recent methods therefore aim to constrain or guide reasoning along the true KG topology. Graph-Constrained Reasoning (GCR) restricts decoding to KG-valid entities and relations via a KG-Trie, reducing hallucinated path steps Luo et al. (2024). MemQ stabilizes reasoning with explicit intermediate memory Xu et al. (2025). Iterative exploration frameworks, e.g. KnowledgeNavigator Guo et al. (2024), Walk&Retrieve Böckling and Paulheim (2025), Think-on-Graph Sun et al. (2023), and decompositional graph retrieval Six et al. (2025), progressively navigate and filter the relevant KG regions. Still, these methods verbalize reasoning into unstructured text or rely on heuristics rather than ensuring globally consistent traversal or provable path correctness, leaving multi-hop reasoning vulnerable to cascading errors.

Other research explored agentic approaches to treat LLMs as planners capable of iteratively retrieving information and invoking tools. Multi-stage designs interleave planning, retrieval, and synthesis to support longer reasoning chains and achieve strong performance on complex QA tasks Yue (2025). Yet current KG-augmented agents often generate unconstrained tool calls, leading to hallucinated actions or inefficient graph exploration. More structured control mechanisms, including the KG-Agent's closed-loop planner–executor–memory architecture Jiang et al. (2024) and GNN-guided KG-RAG for improved subgraph selection Liu (2025), strengthen the grounding. Additional research explored hyper-relational KGs Panda et al. (2024) and structured SFT alignment Chen et al. (2025a). Nonetheless, these systems still lack strict guarantees of graph-consistent planning or step-level verifiability, allowing reasoning to diverge from the true KG and preventing deterministic, auditable multi-hop reasoning over large-scale graphs.

Both KG-RAG or agentic attempts fall short of delivering fully reliable KG-based reasoning. They operate either indirectly over the KG where reasoning traces are verbalized back into unstructured text or they depend on heuristically selected subgraphs, which reintroduces the hallucination risks. They enforce structural consistency only locally rather than across full multi-hop paths, meaning that errors early in the chain still propagate. Moreover, none of these methods guarantee completeness of KG traversal or provide deterministic, step-level provenance verification.

3 GraphWalker

Motivated by the limitations of existing KG-augmented LLM approaches, we propose GraphWalker, an iterative multi-agent framework that performs explicit, multi-hop traversal over a KG. Instead of retrieving large subgraphs upfront or reasoning only indirectly through unstructured text, GraphWalker incrementally explores the KG one hop at a time, ensuring that every reasoning step is grounded in verifiable edges.

As shown in Fig. 1, the architecture consists of a recurring loop with four components: the KG as the underlying knowledge source, an LLM WalkerAgent that inspects the immediate neighbourhood around the current node and proposes the most question-relevant candidate expansions, an Accumulated Walks dictionary that stores gathered triples as structured contextual evidence and an LLM RephraserAgent that decides whether the current evidence suffices to answer the question or further traversal is required, by generating new subquestions for the WalkerAgent.

Algorithm 1 complements the high-level overview in Figure 1 with detailed pseudocode of the full main control flow. It formalizes how GraphWalker selects a starting node, iteratively extends walks through the KG, accumulates evidence, and recursively processes subquestions until a final answer is produced. To avoid the risk of the agentic behaviors looping indefinitely, we additionally enforce an algorithmic limit on the amount of recursions by introducing a maximum recursion depth parameter.

Before describing each individual component in detail in the following subsections, we first present a running example that illustrates the full cycle end-to-end.

3.1 Running example

Fig. 2 illustrates the execution flow of the GraphWalker approach using the example query: *"Which films does the director of [Harry Potter and the Prisoner of Azkaban] also direct?"*. Here, *Harry Potter and the Prisoner of Azkaban* will be abbreviated as *HP PoA* in the remainder of this example. The process begins in (A), where the GraphWalker positions the system at the node `[HP PoA]` within the KG (in this example use case, an arbitrary one). This is determined by the semantics of the original question and will be further explained in Section 3.2. In (B), the

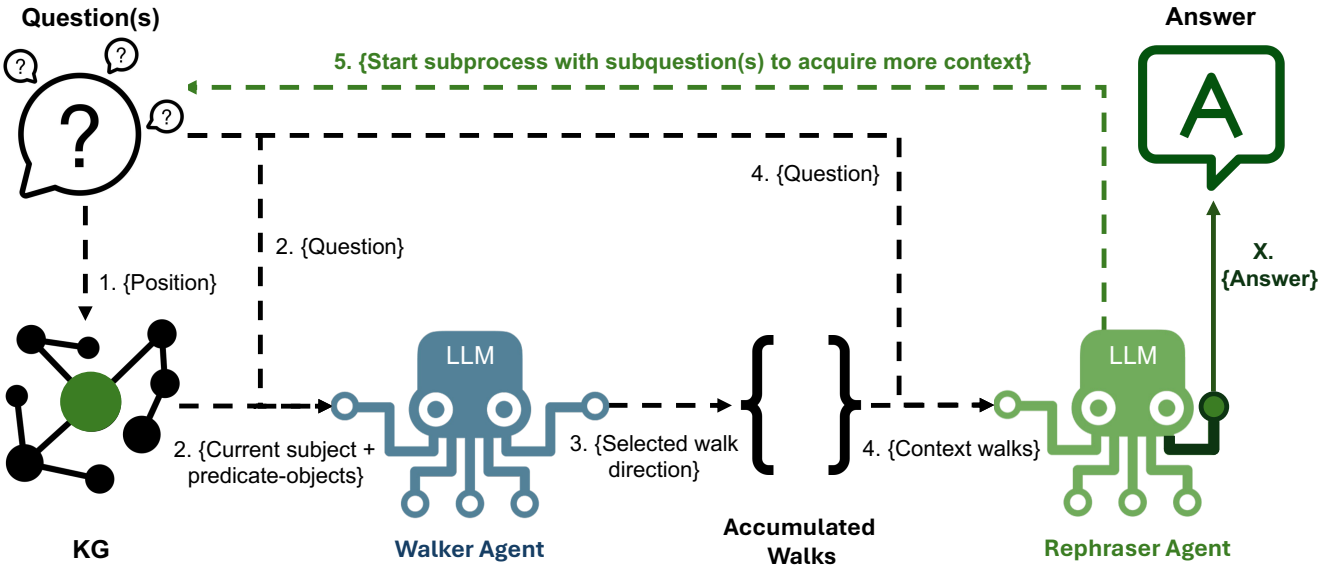

Figure 1. Overview of the GRAPHWALKER approach. Given a question, (1) a starting node is selected within the KG, and (2) both the question and the neighbourhood of this node are provided to the Walker agent. The Walker agent (3) chooses which path to follow to acquire more information relevant to the question. All traversed walks are stored as context (4) and passed to the Rephraser agent. The Rephraser agent decides whether to stop (X) and return an answer to the question or (5) to initiate a new Walker–Rephraser subprocess based on generated sub-question(s) in order to collect additional relevant context from the KG. If the recursion depth is exhausted without an explicit stop decision, (6) a final answer is generated from the available accumulated context.

Algorithm 1: GRAPHWALKER Question Answering

Input: Question q, KG $\mathcal{G}$, Context C (initially $\emptyset$), max recursion depth $\mathcal{D}$
Output: Answer a

Function `GraphWalker`$(q, \mathcal{G}, C, \mathcal{D})$**:**

```
// 1. Identify best starting node in the KG
v ← GraphPositioner(q, G)
// 2. Retrieve neighbourhood
N_v ← Neighbourhood(v, G)
// 3. Walker agent chooses path step
w ← WalkerAgent(q, v, N_v)
C ← C ∪ {w}
while D is not 0 do
    // 4. Rephraser evaluates current context
    (stop, a, Q_sub) ← RephraserAgent(q, C)
    if stop = True then
        return a   // Final answer produced
    // 5. Recursively process generated sub-question(s)
    foreach q_sub ∈ Q_sub do
        a_sub ← GraphWalker(q_sub, G, C, D − 1)
        C ← C ∪ {a_sub}   // insert enriched context
// 6. Fallback: generate final answer from accumulated context
a ← FinalAttemptToAnswer(q, C)
return a
```

Walker Agent receives both the question and the immediate neighbourhood of the current node, offering three potential traversal options to acquire more relevant information: `HP PoA` → `Directed_by` → `Alfonso Cuarón`, `HP PoA` → `Release_year` → `2004`, and `HP PoA` → `Has_genre` → `Fantasy`. Based on relevance to the question, the Walker selects the most promising direction, i.e. `HP PoA` → `Directed_by` → `Alfonso Cuarón`.

In (C), the accumulated path `[HP PoA → Directed_by → Alfonso Cuarón]` is provided to the Rephraser Agent to evaluate whether sufficient context exists to answer the question. Since the system now knows *who* directed `[HP PoA]`, but not *which other films* that director has worked on, the Rephraser generates a sub-question: *"Which movies did Alfonso Cuarón direct?"*. Within this subquestion, the rephraser still has to define the correct entity to proceed. A refinement prompt is used in (D) to extract the correct entity out of the text, given the current entities within our accumulated walks datastructure. Once the entity is identified `[Alfonso Cuarón]` within our subquestion , the new Walker–Rephraser subprocess is started. The Walker receives a new set of traversal options, starting from Alfonso Cuarón and going towards: `Gravity`, `Children of Men`, and `Roma`. To answer the current question, the Walker selects the first two options, which are semantically aligned with the predicate `directed`. In (E), the accumulated walks now consist of:

```
{ [HP PoA → Directed_by → Alfonso Cuarón]
  [Alfonso Cuarón → directed → Gravity]
  [Alfonso Cuarón → directed → Children of Men] }
```

The Rephraser Agent reassesses and determines that the collected context is now sufficient to resolve the original question. Therefore, in (F), the system terminates the iterative

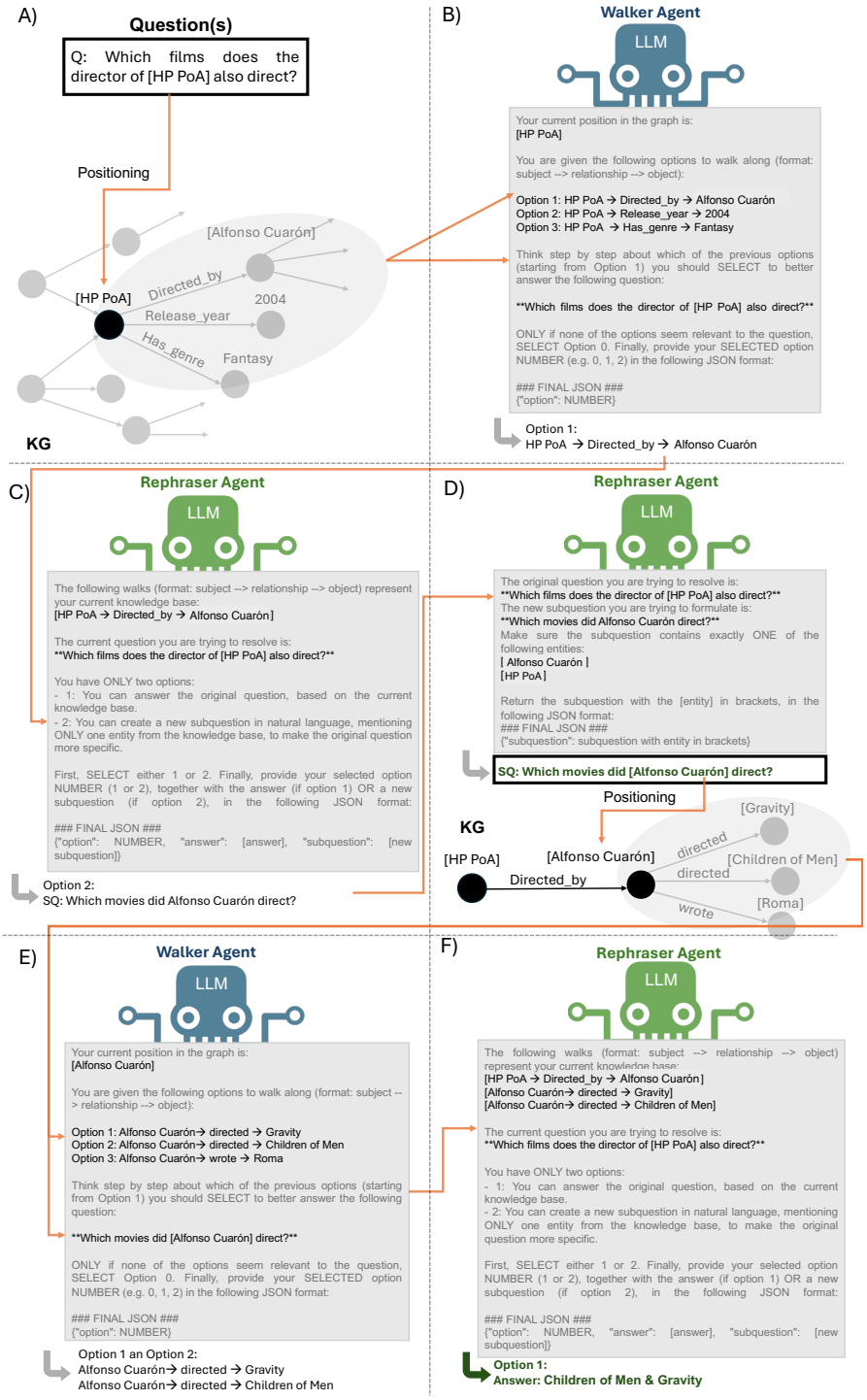

Figure 2. Running example illustrating the execution flow of the GRAPHWALKER

process and returns the final answer: *Gravity* and *Children of Men*.

3.2 Graph positioner

Given the natural language question q and the structure of the KG, the objective of GRAPHPOSITIONER is to identify the most relevant entity in the question that should serve as the starting point for multi-hop exploration. This step is important, as the quality of the selected context, and thus the reasoning process, depends on the starting point of the walk within the graph.

For the purposes of this research, we assume that the relevant entity is already explicitly marked within the question using square brackets. For example, a question such as:

> *"Which films does the director of [Harry Potter and the Prisoner of Azkaban] also direct?"*

clearly identifies `Harry Potter and the Prisoner of Azkaban` as the starting node. In such cases, extracting the initial entity becomes a straightforward string parsing task, and the GRAPHPOSITIONER simply matches this surface form to the corresponding KG node.

This assumption allows us to focus on the core contribution of GraphWalker, i.e. if a relevant starting node can be pinpointed, how well can agentic AI steer the exploration of the graph to find the appropriate answer to the question. Determining the correct entity associated with a question is itself an active research area involving entity recognition, disambiguation, and linking techniques Guellil et al. (2024). Sophisticated approaches typically must resolve ambiguities (e.g., "Paris" the city vs. "Paris" the mythological figure). Such state-of-the-art algorithms could easily be integrated in the GraphPositioner component in future work.

3.3 Walker agent

The WalkerAgent is responsible for choosing the next edge to follow in the KG at each intermediate step. Conceptually, it implements the transition function of the walk: given the current position node and local neighbourhood around that node of depth 1, it selects exactly one outgoing edge or decides to stop exploring from this node. Concretely, this agent receives the following inputs:

- The natural language question q.
- The current position node within the graph.
- A list of *candidate options* `options_text`, each corresponding to a possible outgoing edge from the current node, numbered as Option 1, Option 2, . . . and rendered in the format *subject* → *relationship* → *object*.

Based on these inputs, the Walker agent is prompted to reason step by step about which candidate edge is most useful for making progress toward answering the question q. If none of the options are relevant, it can instead select a special "Option 0" that signals that the current branch should not be extended further.

The core prompt template used for this agent is shown in Listing 1. Here, `{current_node_position}`, `{options_text}` and `{question}` are programmatically filled in at runtime:

Listing 1: Prompt used for the Walker agent.

```
Your current position in the graph is:
{current_node_position}

You are given the following options to walk along (format:
subject --> relationship --> object):
{options_text}

Think step by step about which of the previous options (
starting from Option 1) you should SELECT to better answer the
 following question:
**{question}**

ONLY if none of the options seem relevant to the question,
SELECT Option 0.

Finally, provide your SELECTED option NUMBER (e.g. 0, 1, 2) in
 the following JSON format:

### FINAL JSON ###
{"option": NUMBER}
```

Providing the current traversed path within the graph reminds the LLM of the taken trajectory and thereby grounds the decision in an accumulated traversal history. We use chain-of-thought (CoT) prompting Wei et al. (2022) by explicitly stating in the instruction to "think step by step" before delivering the final output to encourage the agent to compare options with respect to their relevance to the question. The requirement to respond in a strict JSON schema ensures that the output is easy to parse and integrate into the control loop of Algorithm 1.

In practice, we expect the model to (i) implicitly evaluate which entities and relations are semantically aligned with the question, (ii) select the most promising option index, and (iii) return a JSON object of the form `{"option": k}` that deterministically drives the next hop in the GraphWalker reasoning process.

3.4 Rephraser agent

The RephraserAgent is responsible for deciding whether the information gathered so far is sufficient to answer the question, or whether additional traversal within the KG is required. Concretely, the Rephraser agent receives the following inputs:

- The natural language question q.
- A summary of the context gathered so far, derived from the Accumulated Walks dictionary data structure, which linearizes the traversed triples into a sequence of walks (e.g., $\text{Entity}_1 \rightarrow \text{relation}_1 \rightarrow \text{Entity}_2$).

Based on these inputs, the RephraserAgent defines two outcome options: it can either answer the original question directly based on the information in its context window, or propose a new subquestion that focuses on a single entity from the current knowledge base, thus narrowing down the search space for subsequent traversals. The core prompt template used for this decision is shown in Listing 2. Here, `{current_walks}`, `{question}` are populated at runtime:

Listing 2: Prompt template used for the Rephraser agent.

```
The following walks (format: subject --> relationship -->
object) represent your current knowledge base:
{current_walks}

The current question you are trying to resolve is:
**{question}**

You have ONLY two options:
- 1: You can answer the original question, based on the
current knowledge base.
- 2: You can create a new subquestion in natural language,
mentioning ONLY one entity from the knowledge base, to make
the original question more specific.

First, SELECT either 1 or 2.

Finally, provide your selected option NUMBER (1 or 2),
together with the answer (if option 1) OR a new subquestion (
if option 2), in the following JSON format:
### FINAL JSON ###
{"option": NUMBER, "answer": [answer], "subquestion": [new
subquestion]}
```

By explicitly presenting the traversed walks as a "current knowledge base", the prompt reminds the LLM which evidence is available and constrains its reasoning to verifiable triples. The requirement to first select between option 1 (answer) and option 2 (subquestion) makes the stopping decision explicit. The use of CoT through the instruction to "think step by step" encourages the model to relate each walk to the question before committing to an answer or a refinement. The strict JSON schema ensures that the output

is easy to parse: if `"option": 1` is returned, the `"answer"` field is used as the model's final answer; if `"option": 2` is selected, the `"subquestion"` field provides a new query that will be handled recursively by Algorithm 1.

When the Rephraser chooses to generate a new subquestion (option 2), we additionally apply a refinement prompt to ensure that the subquestion is tightly anchored to a single entity from the knowledge base and that this entity is explicitly marked to aid the GraphPositioner, as discussed in Section 3.2. This refinement is defined by the template in Listing 3, which operates on the following inputs:

- The original question q.
- The subquestion q'.
- A list of candidate entities derived from the Accumulated Walks dictionary data structure.

Listing 3: Refinement prompt for constraining subquestions to a single entity.

```
The original question you are trying to resolve is:
**{original_question}**

The new subquestion you are trying to formulate is:
**{subquestion}**

Make sure the subquestion contains exactly ONE of the
following entities:
{entities_text}

Return the subquestion with the [entity] in brackets, in the
following JSON format:
### FINAL JSON ###
{"subquestion": subquestion with entity in brackets}
```

The resulting JSON object provides a normalized, machine-readable representation of the subquestion, which can be used to control subsequent calls to the main GraphWalker loop and subsequent WalkerAgent to precisely track which entities drive each branch of the multi-hop process. In rare cases where no suitable reformulation can be found, the current iteration is simply restarted.

3.5 Termination and fallback answering

To prevent unbounded recursive exploration, we limited GraphWalker to a maximum recursion depth $\mathcal{D}$ in Algorithm 1. If that maximum recursion depth is reached without an explicit request to answer the main question, GraphWalker triggers a fallback mechanism that invokes the FinalAttemptToAnswer component on the original question q and the final state of the accumulated context C. This component linearizes C into `{context_walks_text}` and uses the following prompt:

Listing 4: Prompt used for the FinalAttemptToAnswer fallback.

```
Please provide an answer to the question: **{question}**
Base your answer on the following information:
    {context_walks_text}
If you are not able to answer the question, please return the
following sentence:
'could not answer this question as no relevant information
about the question was available in the knowledge base'
```

This ensures that the system always terminates and either returns an answer grounded in the collected KG evidence or emits a conservative failure message when no relevant information is available.

4 Experimental setup

The GraphWalker approach is evaluated on the MetaQA dataset Zhang et al. (2018), a large-scale multi-hop KGQA benchmark built on top of the WikiMovies knowledge base. It provides questions whose answers are 1, 2, or 3 hops away from an entity in the underlying movie KG, thereby explicitly testing a system's ability to perform compositional multi-hop reasoning over a fixed graph structure Zhang et al. (2018). The KG underlying the text-based MetaQA split contains approximately 43k entities, 9 relations, and 135k triples. The full dataset comprises more than 400k questions distributed across these 1-, 2-, and 3-hop settings Zhang et al. (2018). In line with prior work Böckling and Paulheim (2025), we use the *text-based* version of MetaQA and report results on all three difficulty levels (1–3 hops). The number of test questions per hop, using the standardized splits reported in recent MetaQA evaluations Li et al. (2025b), is 9,947, 14,872 and 14,274 questions respectively for 1-, 2- and 3-hop.

As outlined in the related work (Section 2), three major categories of algorithms to tackle QA over KG can be identified. As a result, we compare our GraphWalker against the following three baselines, each representing one of these categories:

- *The parametric LLM* answers the question directly without any external knowledge and thus capturing purely parametric knowledge.
- *The RAG LLM* baseline embeds each KG triple as a standalone document using the well-stated nomic-embed-text open-source embedding model Nussbaum et al. (2024) and stores it in a Chroma vector store chr (2025). The question is embedded with the same embedder model, and, in line with the current state-of-the-art, the top-3 most similar documents/triples are retrieved from the vector store. These retrieved contexts are then appended to the original prompt to condition the LLM during answer generation.
- *Walk&Retrieve* is a top-performing two-stage pipeline enabling zero-shot KG augmented RAG that builds an offline text corpus by performing random or breadth-first walks around each KG entity, verbalizing each walk into a natural-language sentence with an LLM, and indexing the resulting verbalized walks. At query time, the question is encoded in the same embedding space, the most similar entities and their verbalized walks are retrieved, and an LLM generates the answer from this context, with an explicit instruction that defines when the context is insufficient Böckling and Paulheim (2025).

Note that we have only included zero-shot approaches to represent each of the categories, as GraphWalker itself enables a form of zero-shot RAG that is not tied to any specific backbone model. All methods are evaluated under the unified prompting and decoding setup of the open-source Llama-3.1-70B Dubey et al. (2024) Q4_K_M quantized using the Ollama interface oll (2025) and setting the temperature to zero, similar to previous MetaQA evaluations Böckling and Paulheim (2025). This ensures that differences in performance can be attributed primarily to the retrieval and reasoning strategies rather than to low-level generation

details. The max recursion depth of the GraphWalker approach is set to 4 as the questions only go 3 hops deep at most. We evaluate all systems along three complementary axes:

- **Accuracy**: fraction of questions for which the model prediction matches any of the ground-truth answers.
- **Abstentions**: fraction of questions where the model explicitly refused to provide an answer and also clearly states this within the prompt output.
- **Hallucinating**: fraction of answers where the model confidently states content that is not supported by any ground-truth answer or KG evidence & is thus incorrect.

To acquire these metrics, we do not compare predictions to ground truth using exact string matching, as this might unfairly punish answers that are semantically correct, but syntactically formulated in another fashion than the ground truth answers. Instead, we used a separate LLM as a judging model Li et al. (2025a), according to the methodology used in the Walk&Retrieve paper. This LLM model receives the original question, the obtained predicted answer, and the list of ground-truth answers and is instructed to output a verdict whether or not the prediction matches the ground-truth answers Böckling and Paulheim (2025). For these LLM-as-a-judge evaluations, we used the more recent and powerful Llama-3.3-70B model, again with the temperature set to zero.

5 Results & Discussion

The results are shown in Table 1. The first column lists the various evaluated models, repeated for each question set, i.e., Hop 1, Hop 2, and Hop 3. The other three columns correspond to each of the selected evaluation metrics, calculated over the respective question sets and expressed as percentages. Overall, the results show that GraphWalker delivers substantial gains over all baselines across the three hop depths and align with the requirements outlined in the introduction. In terms of accuracy, GraphWalker consistently outperforms the best-performing baseline, with percentage point improvements of 11.61%, 18.35% and 32.66% for Hop 1–3, respectively. These gains are particularly noteworthy in this multi-hop setting, where both the parametric LLM and the RAG LLM degrade rapidly. At Hop 2, for instance, the RAG LLM drops to 7.48% accuracy with nearly 60% abstentions, whereas GraphWalker retains 81.23% accuracy with 10.34% abstentions. Compared to Walk&Retrieve, the most structurally similar baseline, GraphWalker combines higher accuracy with fewer hallucinations (e.g., 95.24% vs. 78.07% accuracy and 3.34% vs. 21.00% hallucinations on Hop 1), suggesting that its explicit, step-wise control over KG exploration enables a more effective use of the graph.

The combination of accuracy, abstentions and hallucinations is particularly informative for factual grounding. The parametric LLM frequently answers confidently, but incorrectly, with hallucinations dominating its error profile across all 3 hops. The RAG LLM exhibited the opposite pattern on the harder questions. It substantially increases abstentions (up to 31.64% on Hop 3), indicating that it often fails to find or use sufficient evidence to commit to an answer. GraphWalker, by contrast, maintains comparatively low hallucination and abstention rates.

Factual grounding in GraphWalker is further enforced by constraining each intermediate step to explicit KG edges: if a required edge is absent, the system cannot "invent" a relation, but must either explore a different neighbourhood or abstain. Since MetaQA provides a ground-truth answer for every question, any abstention reflects a failure to construct a fully supported path. The combination of low abstention rates and reduced hallucinations therefore indicates that GraphWalker typically either discovers a complete KG-backed reasoning chain or explicitly refrains from answering, reducing the space of unsupported inferences.

However, this observation requires further qualification. On Hop 2, the results evince a favourable balance between abstentions and hallucinations, with with abstentions occurring more frequently than hallucinations, thus supporting the aforementioned contention. However, on Hop 3, the proportionality is reversed, as abstentions are outweighed by hallucinations. This suggests that as questions become more complex, GraphWalker more frequently does not discover a complete KG-backed reasoning chain (the reasoning chain might only be partial with respect to the evidence required by the question), but at the same time fails to abstain. Because the drop in accuracy here is compensated mostly by hallucinations rather than abstentions, it is possible that GraphWalker tries to reconstruct answers based on incomplete reasoning chains. In other words, while comparatively it is able to find coherent reasoning chains more often than its counterparts, it seems it cannot in the final step always reliably decide whether a complex body of evidence actually covers the question at hand. This in turn suggests shortcomings in terms of reasoning ability when confronted with more complex partial evidence chains, something which is potentially exacerbated by the fallback mechanism outlined in Subsection 3.5. As it stands, more research would be required to properly identify the failure modes in question.

The same mechanisms that promote factual grounding also directly support traceability. Because GraphWalker performs 1-hop deterministic expansions and maintains a structured evidence buffer of visited nodes, edges and intermediate answers, it exposes a provenance trace for each prediction. For aggregation-heavy queries such as *"When did the films written by the writers of Send Me No Flowers release?"*, the buffer records both the intermediate entities (writers, related films) and the relations connecting them (e.g., `written_by`, `wrote`, `release_year`), together with the subquestions that triggered each step. This enables users to inspect which KG facts were used, which subquestions were asked, and where potential missing or erroneous edges may lie. Such fine-grained provenance is not readily available in the parametric LLM and is only partially recoverable in the RAG LLM and Walk&Retrieve baselines, where intermediate reasoning is often compressed into similarity scores or free-text summaries.

The hop-wise breakdown also highlights differences in multi-hop reasoning. While all models deteriorate with depth, the parametric LLM remains stuck at low-to-moderate accuracy with high hallucination rates. The RAG LLM collapses on Hop 2 before partially recovering on Hop 3,

Table 1. Performance per hop and model (Accuracy, Abstentions, and Hallucinations in % of the total number of test questions).

	Model	Accuracy ↑ (%)	Abstentions ↓ (%)	Hallucinations ↓ (%)
Hop 1	Parametric LLM	68.35	4.03	27.62
	RAG LLM	83.63	7.85	8.52
	Walk&Retrieve (BFS depth 4)	78.07	**0.92**	21.00
	GraphWalker	**95.24**	1.40	**3.34**
Hop 2	Parametric LLM	37.04	10.60	52.35
	RAG LLM	7.48	59.75	32.77
	Walk&Retrieve (BFS depth 4)	62.88	**3.28**	33.84
	GraphWalker	**81.23**	10.34	**8.43**
Hop 3	Parametric LLM	41.05	4.72	54.22
	RAG LLM	27.43	31.64	40.92
	Walk&Retrieve (BFS depth 4)	43.24	5.00	51.76
	GraphWalker	**75.90**	**3.05**	**21.04**

Table 2. Ablation studies per hop and setting.

	Setting	Accuracy ↑ (%)	Abstentions ↓ (%)	Hallucinations ↓ (%)
Hop 1	Default	95.3	1.9	2.8
	Llama-3.1-8B	93.0	1.5	5.5
	Llama-3.3-70B	95.2	1.1	3.7
	Qwen-3-8B	93.0	4.0	3.0
	Qwen-3-32B	94.1	1.9	4.0
	Multiple Choice	95.4	1.6	3.0
	Simple Prompting	95.5	1.1	3.4
	Positioning	83.1	8.3	8.6
	Filtering	95.5	1.5	3.0
Hop 2	Default	81.6	10.1	8.3
	Llama-3.1-8B	58.2	16.1	25.7
	Llama-3.3-70B	87.9	7.8	4.3
	Qwen-3-8B	44.2	49.7	6.1
	Qwen-3-32B	80.1	15.8	4.1
	Multiple Choice	77.2	11.2	11.6
	Simple Prompting	74.6	12.5	12.9
	Positioning	48.8	24.1	27.1
	Filtering	80.0	10.9	9.1
Hop 3	Default	77.1	2.4	20.5
	Llama-3.1-8B	47.9	12.9	39.2
	Llama-3.3-70B	81.5	5.3	13.2
	Qwen-3-8B	16.0	70.9	13.1
	Qwen-3-32B	74.7	11.0	14.3
	Multiple Choice	70.4	5.1	24.5
	Simple Prompting	62.7	11.8	25.5
	Positioning	58.8	2.5	38.7
	Filtering	72.7	6.6	20.7

a pattern we attribute to the 3-hop questions that loop back to the start node (making the initial retrieved context relevant again) or the fallback to its parametric knowledge. Walk&Retrieve behaves more robustly, but still degrades substantially as chains grow longer. GraphWalker degrades more gracefully, retaining a clear margin over all baselines at Hop 2 and Hop 3, which suggests that treating graph traversal as a first-class, LLM-directed action helps maintain coherent reasoning chains.

6 Ablation Study

To isolate contributions of particular components in the overall design, we have performed various ablation studies. Each of these studies was performed on a small subset of the MetaQA question set (1000 questions were sampled uniformly from each hop set), which was kept invariant across all studies. Overall, the same experimental setup was used as described in Section 4.

Specifically, based on the results displayed in Table 2, we investigated the following contributions:

Influence of Backbone: We evaluated four alternative backbone models, namely, Llama-3.1-8B, Llama-3.3-70B, Qwen-3-8B, and Qwen-3-32B, to gauge the influence both of model size and model type on the performance of GraphWalker. As expected, default behaviour on the smaller subset closely tracks the results outlined in Table 1, as the same setup was used. In terms of accuracy, we find that all models exhibit very similar behaviour for the Hop 1 set, indicating that even the smaller models perform well when the required piece of evidence is available in the neighbourhood of the starting node. When the reasoning complexity increases, however, performance for the smaller models degrades rapidly (e.g., 44.2% for Qwen-3-8B versus 80.1% for Qwen-3-32B on the Hop 2 set, and 16.0% versus 74.7% for the same models on the Hop 3 set). Here, the ability to track evidence chains and meaningfully decompose questions into more refined subquestions based on available evidence becomes a more decisive factor. In terms of the proportion of abstentions vis-à-vis hallucinations, we find that most of the models follow the same overall trend exhibited by the default setting, in that as question complexity increases, the portion of the answers that are hallucinated increases relative to the portion of answers that amount to an explicit abstention. Notably, this trend is more muted in the Qwen models, and especially in the Qwen-3-8B backbone, which maintains relatively low hallucination rates (i.e., the proportion of hallucinations relative to total number of failures) of 10.9% (Hop 2) and 15.6% (Hop 3) versus 61.5% (Hop 2) and 75.2% (Hop 3) for its Llama-3.1-8B counterpart. Compared to the default setting, the Qwen-3-32B backbone exhibits comparable accuracy across all hop sets, while exhibiting consistently more favourable abstention to hallucination ratios.

Influence of Multiple choices: We evaluated whether allowing the Walker agent to select multiple exploratory options during each walk might increase overall performance. While we hypothesized that doing so would decrease exploration complexity as more relevant information could be accumulated per walk, thus potentially decreasing the need for further information aggregation down the line, this did not materialize in the results. Across all hop sets we find that allowing multiple walker options either does not markedly improve performance or marginally decreases accuracy vis-à-vis the default setting. It is possible that adding multiple options merely amounts to a more polluted context, as additional facts mostly contribute noise, thus hindering the backbone model's reasoning ability.

Influence of Prompting verbosity: We also evaluated the influence of the prompt complexity, to verify whether the prompts presented in Section 3 could be made less verbose without significantly degrading performance. Specifically, we removed explicit exhortations to think step by step and made the task descriptions less precise. Regarding the latter, for example, we removed elucidations of the triple format used to represent the knowledge base in both the Walker and Rephraser prompts. Based on the results shown in Table 2, we find that reducing prompt verbosity in this manner, significantly degrades performance on the higher complexity questions, in particular on the Hop 3 set, with drops in accuracy of 7% (Hop 2) and 14.4% (Hop 3) relative to the default setting.

Influence of Positioning: As mentioned in Subsection 3.2, we assume that each question included in MetaQA embeds a relevant starting node, which can then be used to initiate further exploration. The results presented in Table 1 were generated based on this assumption. While throughout this research we have focused on the core contribution of guided graph-based exploration, we also evaluated a rudimentary positioning mechanism as part of the ablation study, to allow GraphWalker to generate one or several potentially relevant starting nodes. To find a starting position, we retrieve the k nearest neighbouring triples with respect to the original query. Based on this initial set, we can construct a preliminary knowledge base that is then referenced when prompting the Rephraser as in Subsection 3.4. By interacting with the Rephraser, GraphWalker can elicit a number of subquestions from the initial query, each of which will be anchored to an entity from the knowledge base after refinement. After this, the reasoning process proceeds as usual. Based on the results in Table 2, we find that positioning (with $k = 5$) degrades accuracy significantly, even for the Hop 1 set, but especially for questions of higher complexity. This underlines the importance of selecting an appropriate starting position, which the current positioning mechanism based on similarity search alone does not guarantee, even if the Rephraser is used to generate relevant starting questions.

Influence of Filtering: Besides initial positioning, GraphWalker includes a second ancillary mechanism that also extends the GraphPositioner. Because subsequent subquestions are increasingly selective, it is likely that not all information that was part of the knowledge base when a given subquestion was formulated, will be needed to resolve it. In other words, just as the core workflow of GraphWalker is built around question refinement by narrowing down the search space in subsequent iterations, we hypothesized that there might also be a benefit to iterative knowledge base refinement, as this narrows down the context space, reducing reasoning complexity for the Rephraser. To perform knowledge base refinement, we employ a filtering mechanism that is analogous to the k nearest neighbourhood search used to generate an initial position for GraphWalker. Instead of matching a query to the most similar triples in the overall dataset, we match new subquestions only to the triples already inside our accumulated knowledge base. By selecting the k accumulated facts that are most closely related to the subquestion currently being resolved, we attempt to reduce context complexity for the Rephraser agent, allowing it to ignore information that is no longer deemed pertinent. Contrary to our hypothesis, the results in Table 2 suggest that filtering (with $k = 25$) has only a minimal or even a slightly detrimental effect on performance. It is possible that, instead of refining the knowledge base to better suit the given question, valuable information is discarded, complicating the reasoning process. As in the case of discovering an appropriate initial position, the overall methodology may yet prove beneficial while an approach that relies exclusively on semantic similarity fails to adequately capture which information needs to be retained during refinement.

7 Conclusion

Many existing RAG approaches that already rely on KGs to enhance retrieval-based QA fail to make full use of graph-structured exploration, thereby allowing hallucination risks to propagate along the reasoning chain. By contrast, GraphWalker enables a form of KG-augmented RAG that explicitly grounds the reasoning and retrieval process in the structure of the KG. Doing so addresses some of the most pressing requirements of LLM-based QA, including better factual grounding of answers in structured resources, traceability of answers via step-level provenance, and a more robust form of multi-hop reasoning by means of targeted explorations. GraphWalker was evaluated on the large multi-hop MetaQA benchmark dataset, where it was found to outperform baselines especially in terms of QA accuracy and hallucinations on all question sets, but also evinced the lowest number of abstentions on the Hop 3 question set. While the GraphWalker's ability to enforce graph-constrained exploration has led to a combination of low abstention rates and hallucinations, ensuring reliable abstentions in case of retrieval failure remains challenging. Additionally, GraphWalker's current design does not incorporate a sophisticated starting point selection mechanism, limiting its generalizability. Our initial results suggest that starting point selection based on semantic similarity alone might not be sufficient to guarantee stable reasoning performance. Future research could focus on enhancing the GraphPositioner with state-of-the-art algorithms to correctly identify the most relevant graph entities associated with a given question.

Code Availability. Both the code of the GraphWalker and the evaluation methodology, together with the evaluation data, have been made available in a separate Github repository: https://github.com/KNowledgeOnWebScale/GraphWalker.

Use of Generative AI. We used the generative AI tools ChatGPT (OpenAI) and Copilot (Microsoft) to assist in editing some text and the preliminary figure captions. All AI-generated content was reviewed and edited by the authors.

Competing interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

(2025) Chroma - the open-source embedding database. URL https://github.com/chroma-core/chroma.

(2025) Ollama. URL https://ollama.com/library/llama3.1:70b.

Acharya DB, Kuppan K and Divya B (2025) Agentic ai: Autonomous intelligence for complex goals–a comprehensive survey. *IEEe Access* .

Afroogh S, Akbari A, Malone E, Kargar M and Alambeigi H (2024) Trust in ai: progress, challenges, and future directions. *Humanities and Social Sciences Communications* 11(1): 1–30.

Amayuelas A, Sain J, Kaur S and Smiley C (2025) Grounding llm reasoning with knowledge graphs. *arXiv preprint arXiv:2502.13247* .

Arslan M, Ghanem H, Munawar S and Cruz C (2024) A survey on rag with llms. *Procedia computer science* 246: 3781–3790.

Brown TB, Mann B, Ryder N, Subbiah M, Kaplan J, Dhariwal P, Neelakantan A, Shyam P, Sastry G, Askell A et al. (2020) Language models are few-shot learners. In: *Advances in Neural Information Processing Systems (NeurIPS)*.

Böckling M and Paulheim H (2025) Walk&retrieve: Simple yet effective zero-shot retrieval-augmented generation via knowledge graph walks. URL https://arxiv.org/abs/2505.16849. Preprint arXiv:2505.16849.

Ceccarelli C, Raganato A and Viviani M (2025) Knowledge-grounded detection of factual hallucinations in large language models.

Chakraborty A (2024) Multi-hop question answering over knowledge graphs using large language models. *arXiv preprint arXiv:2404.19234* .

Chen H, Shen X, Wang J, Wang Z, Lv Q, He J, Wu R, Wu F and Ye J (2025a) Knowledge graph finetuning enhances knowledge manipulation in large language models. In: *International Conference on Learning Representations*, volume 2025. pp. 91387–91409.

Chen Y, Sun S and Hu X (2025b) DRKG: Faithful and interpretable multi-hop knowledge graph question answering via llm-guided reasoning plans. *Applied Sciences* 15(12): 6722. DOI: 10.3390/app15126722.

Dubey A, Jauhri A, Pandey A, Kadian A, Al-Dahle A, Letman A, Mathur A, Schelten A, Yang A, Fan A et al. (2024) The llama 3 herd of models. *arXiv e-prints* : arXiv–2407.

Farquhar S, Kossen J, Kuhn L and Gal Y (2024) Detecting hallucinations in large language models using semantic entropy. *Nature* 630(8017): 625–630.

Guellil I, Garcia-Dominguez A, Lewis PR, Hussain S and Smith G (2024) Entity linking for english and other languages: a survey. *Knowledge and Information Systems* 66(7): 3773–3824.

Guo T, Yang Q, Wang C, Liu Y, Li P, Tang J, Li D and Wen Y (2024) Knowledgenavigator: Leveraging large language models for enhanced reasoning over knowledge graph. *Complex & Intelligent Systems* 10(5): 7063–7076.

Han H, Wang Y, Shomer H, Guo K, Ding J, Lei Y, Halappanavar M, Rossi RA, Mukherjee S, Tang X et al. (2024) Retrieval-augmented generation with graphs (graphrag). *arXiv preprint arXiv:2501.00309* .

Herrera F (2025) Making sense of the unsensible: Reflection, survey, and challenges for xai in large language models toward human-centered ai. *arXiv preprint arXiv:2505.20305* .

Huang G, Long Y and Luo C (2025a) Improving multi-hop question answering with prompting explicit and implicit knowledge aligned human reading comprehension. *International Journal of Machine Learning and Cybernetics* : 1–16.

Huang Y et al. (2025b) A survey on hallucination in large language models. *ACM Computing Surveys* Comprehensive taxonomy of hallucination types, causes, detection, and mitigation.

Jiang J, Zhou K, Zhao WX, Song Y, Zhu C, Zhu H and Wen JR (2024) KG-Agent: An efficient autonomous agent framework for complex reasoning over knowledge graph. *arXiv preprint arXiv:2402.11163* .

Kazlaris I et al. (2025) From illusion to insight: A taxonomic survey of hallucinations in large language models. *AI* 6(10).

Kim B and Yang J (2025) Enhancing traceability and interpretability of datasets for rag evaluation: A context-id-aware and graph-based visualization approach. In: *International Conference on Multimedia Information Technology and Applications*.

Springer, pp. 141–152.

Kümmel M (2025) Ragtruth++: Enhanced hallucination detection benchmark. URL https://www.blueguardrails.com/en/blog/ragtruth-plus-plus-enhanced-hallucination-detection-benchmark. Blue Guardrails.

Lavrinovics E, Biswas R, Bjerva J and Hose K (2025) Knowledge graphs, large language models, and hallucinations: An nlp perspective. *Journal of Web Semantics* 85: 100844.

Lecue F (2020) On the role of knowledge graphs in explainable ai. *Semantic Web* 11(1): 41–51.

Li D, Jiang B, Huang L, Beigi A, Zhao C, Tan Z, Bhattacharjee A, Jiang Y, Chen C, Wu T et al. (2025a) From generation to judgment: Opportunities and challenges of llm-as-a-judge. In: *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*. pp. 2757–2791.

Li X, Gao Y, Zhan Z and Liu H (2025b) A large-model-based dual-stage reasoning qa approach with multi-hop path selection. *Journal of Artificial Intelligence Research* 2(2): 1–11.

Liu J (2025) Think parallax: Solving multi-hop problems via multi-view knowledge-graph-based retrieval-augmented generation. *arXiv preprint arXiv:2510.15552* .

Lu C, Luo J, Yu K et al. (2025) Towards explainable language reasoning via multi-modal knowledge graphs. DOI:10.21203/rs.3.rs-7677808/v1. URL https://doi.org/10.21203/rs.3.rs-7677808/v1. Preprint (Version 1).

Luo L, Zhao Z, Haffari G, Li YF, Gong C and Pan S (2024) Graph-constrained reasoning: Faithful reasoning on knowledge graphs with large language models. *arXiv preprint arXiv:2410.13080* .

Ma C, Chen Y, Wu T, Khan A and Wang H (2025) Unifying large language models and knowledge graphs for question answering: Recent advances and opportunities. In: *Proceedings of the 28th International Conference on Extending Database Technology (EDBT)*.

Mathew DE, Ebem DU, Ikegwu AC, Ukeoma PE and Dibiaezue NF (2025) Recent emerging techniques in explainable artificial intelligence to enhance the interpretable and understanding of ai models for human. *Neural Processing Letters* 57(1): 16.

Matton K, Ness RO, Guttag J and Kıcıman E (2025) Walk the talk? measuring the faithfulness of large language model explanations. *arXiv preprint arXiv:2504.14150* .

Mavi V, Jangra A, Jatowt A et al. (2024) Multi-hop question answering. *Foundations and Trends® in Information Retrieval* 17(5): 457–586.

Min S, Wallace E, Singh S, Gardner M, Hajishirzi H and Zettlemoyer L (2019) Compositional questions do not necessitate multi-hop reasoning. *arXiv preprint arXiv:1906.02900* .

Nussbaum Z, Morris JX, Duderstadt B and Mulyar A (2024) Nomic embed: Training a reproducible long context text embedder. *arXiv preprint arXiv:2402.01613* .

Ouyang L, Wu J, Jiang X, Almeida D, Wainwright C, Mishkin P, Zhang C, Agarwal S, Slama K, Ray A et al. (2022) Training language models to follow instructions with human feedback. In: *Advances in Neural Information Processing Systems (NeurIPS)*.

Pan JZ, Razniewski S, Kalo J, Singhania S, Chen J, Dietze S, Jabeen H, Zhang W, Lissandrini M, Biswas R, de Melo G, Bonifati A, Vakaj E, Dragoni M and Graux D (2023) Large language models and knowledge graphs: Opportunities and challenges. *arXiv preprint arXiv:2308.06374* .

Pan S, Luo L, Wang Y, Chen C, Wang J and Wu X (2024) Unifying large language models and knowledge graphs: A roadmap. *IEEE Transactions on Knowledge and Data Engineering* 36(7): 3580–3599.

Panda P, Agarwal A, Devaguptapu C, Kaul M and Ap P (2024) Holmes: Hyper-relational knowledge graphs for multi-hop question answering using llms. In: *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. pp. 13263–13282.

Qiao Z, Li N, Huang C, Wang G, Liang S and Zhao W (2024) GraphLLM: A general framework for multi-hop question answering over knowledge graphs using large language models. In: *Proc. of the CCF Int. Conf. on Natural Language Processing and Chinese Computing (NLPCC)*. pp. 136–148.

Sahoo P, Chinnappa D, Song YI, Cavallari S et al. (2024) A comprehensive survey of hallucination in large language models. In: *Findings of the Association for Computational Linguistics: EMNLP 2024*.

Sağlam BD (2024) *Knowledge Graph Augmented Multi-hop Question Answering Using Large Language Models*. Master's Thesis, Middle East Technical University.

Six V, Dufraisse E and de Chalendar G (2025) Decompositional reasoning for graph retrieval with large language models. URL https://arxiv.org/abs/2506.13380.

Sun J, Xu C, Tang L, Wang S, Lin C, Gong Y, Ni LM, Shum HY and Guo J (2023) Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. URL https://arxiv.org/abs/2307.07697.

Tang Y and Yang Y (2024) Multihop-rag: Benchmarking retrieval-augmented generation for multi-hop queries. *arXiv preprint arXiv:2401.15391* .

Varshney N, Yao W, Zhang H, Chen J and Yu D (2023) A stitch in time saves nine: Detecting and mitigating hallucinations of llms by validating low-confidence generation. In: *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*.

Vladika J, Mülln L and Matthes F (2024) Enhancing answer attribution for faithful text generation with large language models. *arXiv preprint arXiv:2410.17112* .

Vu TH et al. (2024) Refreshing large language models with search engine retrieval. In: *Findings of the Association for Computational Linguistics: ACL 2024*. Introduces the FreshQA benchmark and shows LLMs struggle with up-to-date knowledge without retrieval.

Wei J, Wang X, Schuurmans D, Bosma M, Xia F, Chi E, Le QV, Zhou D et al. (2022) Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems* 35: 24824–24837.

Xu M, Liang G, Chen K, Wang W, Zhou X, Yang M, Zhao T and Zhang M (2025) Memory-augmented query reconstruction for llm-based knowledge graph reasoning. *arXiv preprint arXiv:2503.05193* .

Ye H, Liu T, Zhang A, Hua W and Jia W (2023) Cognitive mirage: A review of hallucinations in large language models. *arXiv preprint arXiv:2309.06794* .

Yue M (2025) A survey of large language model agents for question answering. *arXiv preprint arXiv:2503.19213* .

Zhang J, Zhou W and Ujcich BE (2024a) Provenance-enabled explainable ai. *Proceedings of the ACM on Management of Data* 2(6): 1–27.

Zhang X, Wang M, Yang X, Wang D, Feng S and Zhang Y (2024b) Hierarchical retrieval-augmented generation model with rethink for multi-hop question answering. *arXiv preprint arXiv:2408.11875* .

Zhang Y, Dai H, Kozareva Z, Smola A and Song L (2018) Variational reasoning for question answering with knowledge graph. In: *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

Zhao J, Zhang P, Wang Y, Xin R, Lu X, Li R, Lyu S, Ou Z and Song M (2025) Tqagent: Enhancing table-based question answering with knowledge graphs and tree-structured reasoning. *Applied Sciences (2076-3417)* 15(7).

Zhu T, Liu S, Li B, Liu J, Liu P and Zheng F (2024) Graph reasoning over explicit semantic relation. *High-Confidence Computing* 4(2): 100190.